Trabajo Práctico №1

Ajuste de exponenciales

Abril 2017 — Optimización

Integrante	LU	Correo electrónico
Hurovich, Gustavo Martín	426/13	gushurovich@gmail.com
López, Agustina Florencia	120/13	agustina.lopez@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria – Pabellón I, Planta Baja
Intendente Güiraldes 2160 – C1428EGA
Ciudad Autónoma de Buenos Aires, Rep. Argentina
Tel/Fax: +54 11 4576 3359
http://exactas.uba.ar

Introducción

El objetivo de este trabajo será implementar dos algoritmos para encontrar la función que mejor aproxima a algunos datos observados. Sabemos que originalmente los datos provienen de una exponencial, aunque sujetos a cierto ruido. Por lo tanto, el problema consiste en buscar, dentro de la familia de funciones exponenciales, la que minimiza la distancia con nuestras observaciones. Formalmente, definimos $F = \{f : \mathbb{R} \to \mathbb{R} / f(x) = ke^{ax} + c\}$, y buscamos $min_{f \in F} ||f(x) - y||$

Nuestra herramienta base será *cuadrados mínimos*. Sin embargo, como nuestros problemas son no lineales, tendremos que encontrar la manera de modificarlos acordemente apara efectivamente poder usar el método.

1. Ejercicio 1: Regresión exponencial

1.1. Desarrollo

Como ya mencionamos, buscaremos una función exponencial para aproximar a nuestros datos. En este caso, intentaremos hallar una de la forma $f(x) = ke^{ax}$. Representaremos a nuestros datos mediante dos vectores, $\mathbf{x} = (x_1, \ldots, x_N)$ e $\mathbf{y} = (y_1, \ldots, y_N)$. Si todos ellos provinieran de una exponencial sin errores, tendríamos que $\mathbf{y} = ke^{a\mathbf{x}}$. Como usualmente esto no sucede, vamos a minimizar

$$\sum_i (y_i - ka^{x_i})^2$$

lo que es equivalente a

$$\min_{a,k} \left\| \mathbf{y} - ka^{\mathbf{x}} \right\|, \qquad \text{donde } \; a^{\mathbf{x}} = \left(a^{x_1}, \ldots, a^{x_n}\right). \tag{1}$$

Para poder abordar este problema desde el enfoque de cuadrados mínimos, necesitaríamos linealizarlo. Consideremos entonces la función $g(x_1, \ldots, x_n) = (log(x_1), \ldots, log(x_N))$ y resolvamos

$$\min_{\hat{a} \in \mathbb{R},\ \hat{k} \in \mathbb{R}} \left\| g(\mathbf{y}) - g(ka^{\mathbf{x}}) \right\|,$$

O, equivalentemente,

$$\min_{\hat{a} \in \mathbb{R},\ \hat{k} \in \mathbb{R}} \left\| log(\mathbf{y}) - \hat{k}\mathbb{1} - \mathbf{x}\hat{a} \right\|,$$

donde $\hat{k} = log(k)$ y $\hat{a} = log(a)$.

Si consideramos la matriz

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

lo que queremos encontrar es el vector z = (k̂,â) tal que minimice

$$||Az - b||$$

siendo $b = log(\mathbf{y})$.

Para ello, una vez construida nuestra matriz, planteamos la ecuación $Az = b$ y luego multiplicamos a izquierda por A^t, de donde resulta

$$A^t A z = A^t b$$

Luego, usando el método de QR para la matriz $A^t A$, lo que hallamos es z tal que

$$z = R^{-1} Q^t A^t b$$

Finalmente, a partir de z, reconstruimos k y a aplicando exponenciales respectivamente, y consiguiendo así los valores de la exponencial que mejor aproxima (en el sentido de cuadrados mínimos) a nuestra nube de puntos recibidos como dato.

Es importante notar que los k y a encontrados no necesariamente se corresponden con los que cumplen (1). Sin embargo, en la práctica esto sí sucede, y aún en los casos donde la solución no sea la óptima, probablemente no esté muy lejos de ella. Por lo tanto, no nos preocuparemos por esto.

1.2. Implementación

Recordemos que, para poder aplicar la técnica de cuadrados mínimos, utilizamos la función *log*, cuyo dominio son los números positivos. A la hora de desarrollar nuestro algoritmo, nos topamos con un primer problema: entre nuestros datos del vector **y** había valores positivos y negativos. Una primer idea fue tomar los datos negativos y considerar su valor absoluto, pero esto no parecía ser lo más adecuado, pues en ese caso los valores negativos estarían empujando hacia arriba a nuestra exponencial. Tambien consideramos truncarlos a cero, o a un número positivo muy pequeño, pero notamos que al aplicarles logaritmo pasaban a tener mucho peso. Finalmente decidimos prescindir de esos datos, sabiendo que si el origen de los mismos era una exponencial, podíamos considerarlos anómalos.

Más adelante, notamos que si la exponencial buscada tenía signo negativo, entonces sí tenía sentido que los datos sean menores a cero, por lo que los datos que había que eliminar eran los positivos. Por lo tanto, decidimos tomar como signo de nuestra función al signo del dato de mayor valor absoluto, y eliminar los datos que no coincidían con éste.

En el siguiente gráfico, representamos con puntos azules los datos provenientes de un problema real, y con línea roja la función exponencial conseguida por nuestro algoritmo que se ajusta a ellos.

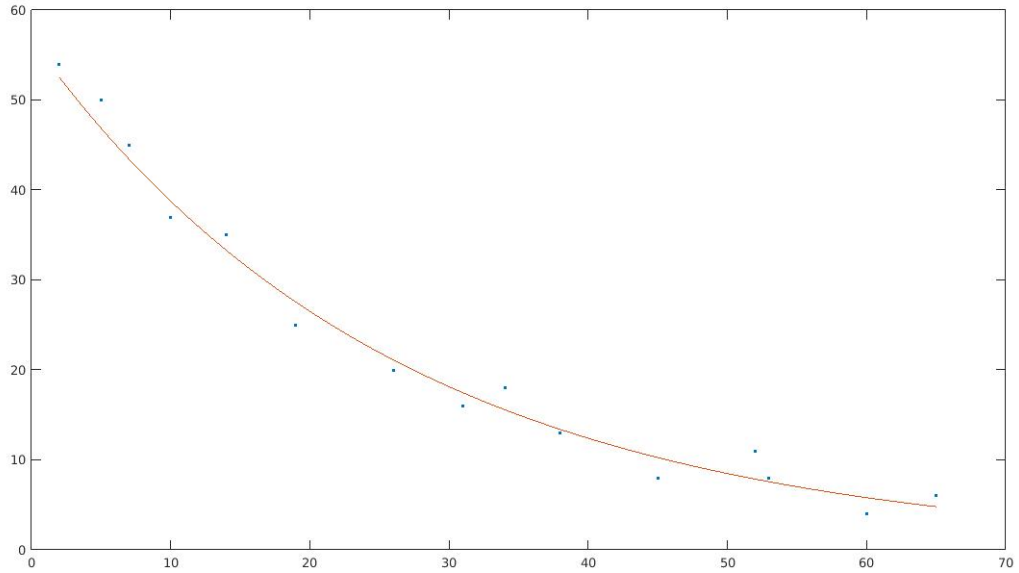

Figura 1: Datos reales y su aproximación exponencial

2. Ejercicio 2: Exponencial generalizada

En este ejercicio, seguimos queriendo conseguir una exponencial que aproxime nuestros datos. Esta vez, vamos a pedir que dicha función sea de la forma $f(x) = ke^{bx} + c$, lo que nos impide usar la estrategia propuesta en el ejercicio 1, o al menos hacerlo de manera directa.

Llamamos

$$d_i(b, k, c) = \left|y_i - ke^{bx_i} - c\right|$$

Y ahora, lo que queremos es encontrar $\theta = (b, k, c)$ que minimiza

$$\sum_{i=1}^{N} \left(y_i - ke^{bx_i} - c\right)^2.$$

Al tratarse de un problema de cuadrados mínimos no lineal, nos vemos obligados a conseguir un método alternativo para poder resolverlo. En este caso, el elegido fue el método iterativo de Gauss-Newton: a partir de una solución inicial $\theta_0 = (b_0, k_0, c_0)$ resuelve el problema de cuadrados mínimos clásico sobre la siguiente ecuación

$$\mathbf{J}(\theta_0)\mathbf{h} = \text{-}\mathbf{d}(\theta_0),$$

construye $\theta_1 = \theta_0 + \mathbf{h}$ e itera. Acá, J(θ) es la matriz diferencial de $\mathbf{d}(\theta)$. En nuestro caso, la condición para finalizar este algoritmo fue alcanzar una cantidad de iteraciones fija, o conseguir que $||\mathbf{h}||$ sea menor a cierta tolerancia.

Para poder programar este método, debimos calcular explícitamente $\mathbf{J}(\theta)$, siendo ésta:

$$\mathbf{J}(\theta) = \begin{bmatrix} -kx_1e^{bx_1} & -e^{bx_1} & -1 \\ -kx_2e^{bx_2} & -e^{bx_2} & -1 \\ \vdots & \vdots & \vdots \\ -kx_Ne^{bx_N} & -e^{bx_N} & -1 \end{bmatrix}$$

En cada iteración, luego de calculada $\mathbf{J}(\theta_i)$, volvemos a usar el comando qr para así conseguir un nuevo vector $\mathbf{h}$ que nos permita avanzar hacia el siguiente paso. Es decir, para encontrar el h que resuelva:

$$R\mathbf{h_i} = Q^t(\text{-}\mathbf{d}(\theta_i))$$

Finalmente, cuando los $\mathbf{h_i}$ son muy pequeños, o realizamos una cantidad grande de iteraciones, el programa para y grafica la función encontrada, con los b, k y c encontrados.

Un problema que nos encontramos fue que nuestro programa funcionaba muy bien cuando la estimación inicial -el θ_0- era realmente cercano al valor real buscado. Sin embargo, cuando esto no sucedía, el programa corría una gran cantidad de iteraciones, o en muchos casos ni siquiera convergía. Cuando el problema es artificial, es decir generado a partir de una exponencial conocida con ruido, podíamos determinar buenos datos iniciales. Pero al analizar los casos reales, con los datos provistos por la materia, no contábamos con dicha estimación.

Decidimos utilizar entonces, lo realizado en el ejercicio 1. Utilizamos dicho algoritmo con nuestros datos, para obtener una estimación inicial de k y b (utilizando para éste último $b = log(a)$). Para el c, decidimos usar el mínimo valor observado de $\mathbf{y}$.

A continuación mostramos tres ejemplos. El primero, una exponencial artificial de parámetros conocidos, con ruido agregado. Los dos restantes, correspondiéndose a dos de los sets de datos provistos.

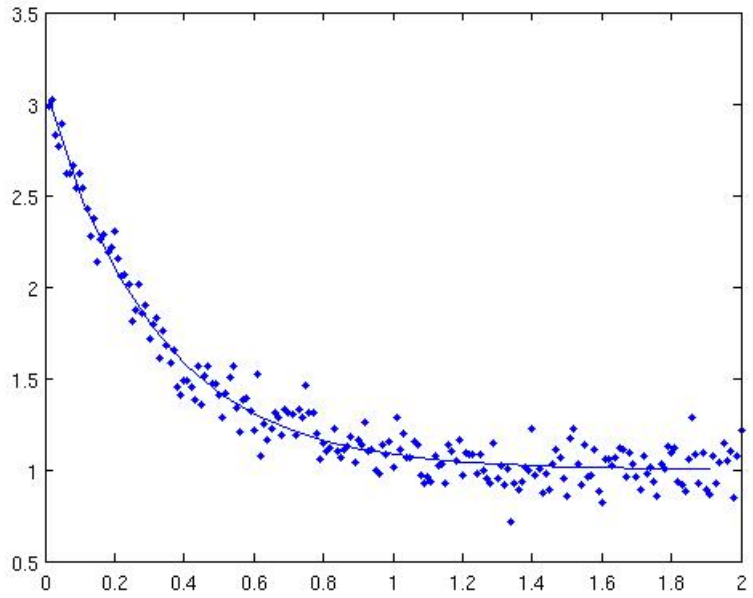

Figura 2: Aproximación de datos. Los puntos azules corresponden a $y = 2 * e^{-3x} + 1$ más un ruido gaussiano. La curva obtenida fue $1{,}97e^{-2{,}81x} + 1{,}00$.

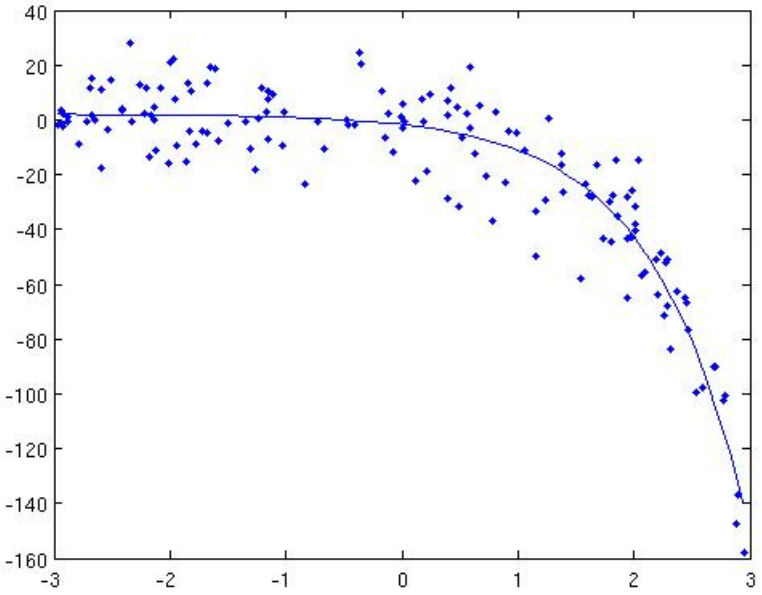

Figura 3: Datos2 del set datosTP1

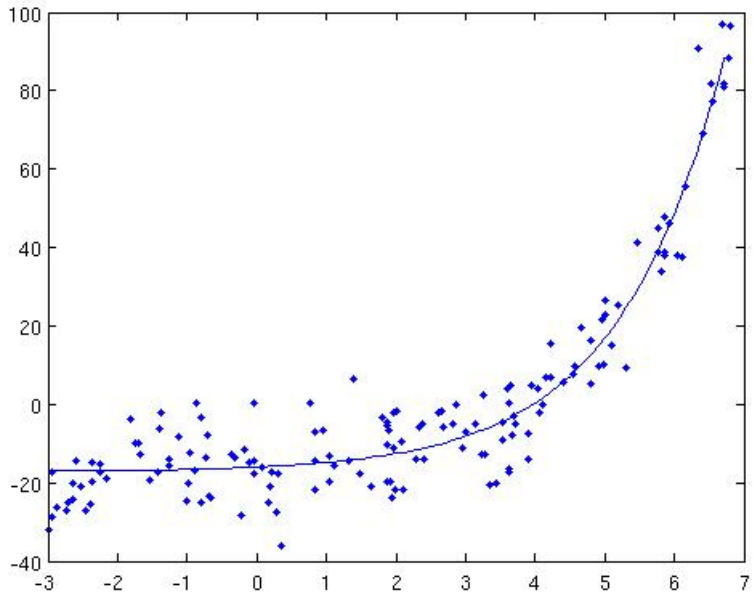

Figura 4: Datos4 del set datosTP1